Distributed Latent Variable Models of Lexical Co-occurrences

John Blitzer
Department of Computer
and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Amir Globerson
Interdisciplinary Center
for Neural Computation
The Hebrew University
Jerusalem 91904, Israel

Fernando Pereira
Department of Computer
and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract

Low-dimensional representations for lexical co-occurrence data have become increasingly important in alleviating the sparse data problem inherent in natural language processing tasks. This work presents a distributed latent variable model for inducing these low-dimensional representations. The model takes inspiration from both connectionist language models [1, 16] and latent variable models [13, 9]. We give results which show that the new model significantly improves both bigram and trigram models.

1 Introduction

Data sparseness is a central problem in machine learning for natural-language processing tasks such as parsing, language modeling, machine translation, and automatic summarization [5, 10, 3]. All these tasks use sparse data to estimate lexical co-occurrence probabilities. One example is modeling the probability $p(w|h)$ that a word w occurs in the context of some other word or words h.

The most common way of dealing with the sparse data problem is to interpolate the full context with shorter contexts, combining the full and shortened statistics into a smoother overall estimate (see [4]). Interpolation models are simple and reasonably effective, but they fail to take into account useful regularities across contexts.

An effective approach to capturing those regularities is to map contexts into low-dimensional representations, and then use those representations to predict w. Two lines of work which have used these ideas are latent variable models and distributed connectionist models.

In latent variable models, one introduces an unobserved discrete random variable with low cardinality, which represents an underlying class of contexts, and separates the context from the word w. The parameters of the model are then estimated via EM or related algorithms [13, 9].

Distributed models map contexts to continuous low dimensional vectors $g(h) \in \Re^n$. The distribution $p(w|h)$ is then modeled as a non linear function of $g(h)$. Recent work by Bengio *et al*. [1] and by Xu *et al*. [16] has shown that this approach yields state of the art performance.

The multidimensional representation of distributed models is attractive, since it can be thought of as representing different, possibly independent, properties of the context. On the other hand, in latent models the intermediate representation has a clear probabilistic interpretation.

In this work, we suggest a model that combines the advantages of the above approaches by using a *vector* of discrete latent variables. Our latent variable model can be nicely described as a directed graphical model, which also encompasses single latent variable models such as the aggregate Markov model (AMM) of Saul and Pereira [13]. Because it uses a vector of latent variables, it also extends naturally to more general n-grams. We show empirically that our model significantly outperforms the AMM on a verb-object bigram modeling task, and we further demonstrate improved performance over a standard baseline for trigram lan-

guage modeling.

The rest of the paper is organized as follows. Section 2 reviews the AMM and introduces the distributed latent variable model. Section 3 describes the EM algorithms for optimizing the model. Section 4 describes our experimental procedures and results. We conclude in Section 6.

2 Distributed Latent Variable Models

In what follows, we describe a model for the probability of observing a word w given a history of previous words h. We begin with a single word history, in order to compare to standard latent models, and then discuss extensions to multivariate histories. The simplest, *non-distributed*, latent model assumes the existence of a low cardinality hidden variable z which separates w and h in the sense that it makes them conditionally independent

$$p(w, h|z) = p(w|z)p(h|z) \quad .$$

The corresponding graphical model is the Bayes net in Figure 1. The conditional probability of w given h is obtained by marginalizing over z:

$$p(w|h) = \sum_z p(z|h) \cdot p(w|z) \quad .$$

The parameters of this model are $p(z|h)$ and $p(w|z)$. The expression for the conditional distribution suggests that z can be thought of as a clustering of h, from which w is then predicted.

Saul and Pereira [13] used this model for language modeling, where they called it the aggregate Markov model (AMM). Hofmann and Puzicha [9] studied it as a special case of what they call "aspect" models. The AMM also falls under the rubric of mixture models. The distribution $p(w|h)$ can be thought of as a mixture of $|z|$ low-dimensional distributions, $p(w|z)$, each of which has mixing weight $p(z|h)$. We will make use of this view of the latent variable model in our analysis.

To use the advantages of a distributed model, we now assume that the latent variable is a *vector* of m binary latent variables (bits)[1] $\mathbf{b} = (b_1, \ldots, b_m)$. As in the previous latent variable model, the latent variables are assumed to separate h and w.

[1]Using binary latent variables simplifies the notation, exposition, and implementation without real loss of generality.

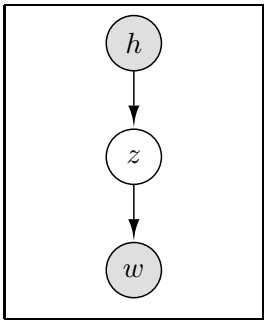


Figure 1: AMM graphical model

A desired property of a distributed model is that its vector elements model independent properties of their inputs. This can be achieved in our model by constraining the variables b_i to be conditionally independent given h. The above requirement can be represented graphically using the Bayes net of Figure 2. To further exploit the distributed nature of $\mathbf{b}$, we use the following multinomial logistic (also known as conditional maximum entropy) model for $p(w|\mathbf{b})$

$$p(w|\mathbf{b}) = \frac{1}{Z(\mathbf{b})} \exp \sum_{i=1}^{m} \psi(b_i, w) \qquad (1)$$

with separate interaction potentials $\psi(b_i, w)$ for each variable b_i. As in standard condtional maximum entropy models, we assume $\psi(b_i, w)$ is an arbitrary, real-valued function of b_i and w.

We refer to the above model as the distributed Markov model (henceforth DMM). The distribution it induces is specified as follows:

$$\begin{aligned} p(w|h) &= \sum_{\mathbf{b}} p(\mathbf{b}|h) \cdot p(w|\mathbf{b}) \\ &= \sum_{\mathbf{b}} \prod_{i=1}^{m} p(b_i|h) \cdot \frac{1}{Z(\mathbf{b})} \exp \sum_{i=1}^{m} \psi(b_i, w) \quad . \end{aligned}$$

The parameters of the model are the binomial parameters $p(b_i|h)$, and the real-valued functions $\psi(b_i, w)$.

The distribution $p(w|h)$ can be seen to be a mixture of 2^m components. However, the DMM has only $O(m)$ parameters. This should be contrasted with the AMM which needs $O(2^m)$ parameters to model a mixture of the same size. When the latent variable has a distributed nature, we expect the DMM model to outperform the standard mixture model, since its fewer pa-

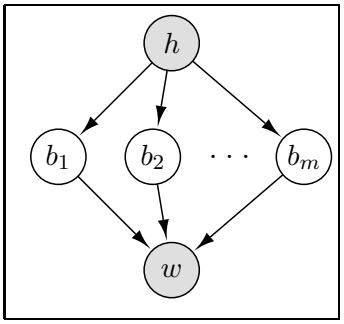


Figure 2: Graphical model corresponding to the distributed latent variable model.

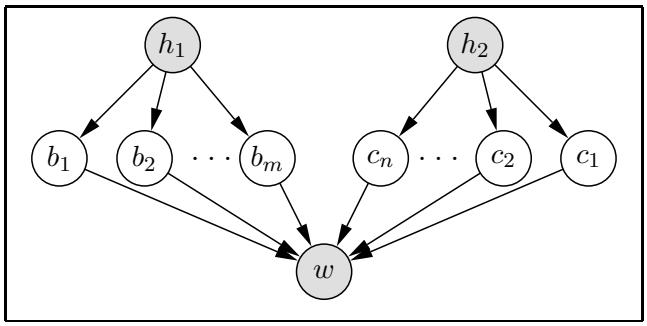


Figure 3: Graphical model corresponding to the multivariate history DMM

rameters make it less likely to overfit. This is indeed seen in the experiments described later.

2.1 Multivariate Histories

The distributed model is easily extended to multi-word histories. For ease of exposition, we discuss here the two-word history (trigram) case. Each word in the history is encoded with a separate vector of latent variables as shown in the graphical model of of Figure 3.

We denote latent variables encoding h_1 and h_2 by $\mathbf{b} = (b_1, \ldots, b_m)$ and $\mathbf{c} = (c_1, \ldots, c_n)$ respectively. The distribution $p(w|\mathbf{b}, \mathbf{c})$ is assumed to factor with respect to the latent variables as in Equation 1. The interaction potentials for $\mathbf{c}$ will be denoted by $\psi(c_i, w)$. The resulting distribution $p(w|h_1, h_2)$ is a mixture of 2^{m+n} elements, with $O(m+n)$ parameters. The vectors $\mathbf{b}$ and $\mathbf{c}$ might have different lengths, allowing us to use more latent variables for more recent history elements, for example.

3 Learning Distributed Models

We now discuss the estimation of the parameters specifying the distributed model, from a given training set. We focus on the two-word history case. Since the model is a mixture distribution, its parameters may be estimated using the expectation-maximization algorithm [7]. Applications of EM to learning single latent variable models are described by Saul and Pereira [13], and Hofmann and Puzicha [9].

In what follows we denote by $\Theta^{(t)}$ the full parameter set at the iteration t of the EM algorithm. Recall that the free parameters are the binomial parameters $p(b_i|h_1), p(c_j|h_2)$ and the functions $\psi(b_i, w), \psi(c_j, w)$. We denote all parameters at time t with a superscript t.

In the E step of the algorithm, posterior probabilities of the latent variables are calculated given the model $\Theta^{(t)}$. For the DMM, this posterior is [2]

$$p^{(t+1)}(\mathbf{b}, \mathbf{c}|w, h, \Theta^{(t)}) \propto p^{(t)}(w|\mathbf{b}, \mathbf{c}) p^{(t)}(\mathbf{b}|h_1) p^{(t)}(\mathbf{c}|h_2)$$

Observe that, because of the DMM's factored form, the posterior can be computed as a normalized product of mixture weights and mixture components. Because of this, the histories h and predicted words w decouple, and we can compute the partition function $Z(\mathbf{b}, \mathbf{c})$ independently of h. Thus the E step for the model requires $O(L \cdot 2^{m+n})$ time, where L refers to the total number of training instances. This can be significantly smaller than the required time for other distributed models.

From this, we construct the joint distribution $p^{(t+1)}(h, w, \mathbf{b}, \mathbf{c})$ over the observed and latent variables

$$p^{(t+1)}(h, w, \mathbf{b}, \mathbf{c}) \propto p^{(t+1)}(\mathbf{b}, \mathbf{c}|h, w, \Theta^{(t)}) p_0(h, w)$$

where $p_0(h, w)$ is the empirical distribution of the observed variables.

The M step uses the above posteriors to estimate a new parameter set $\Theta^{(t+1)}$. The parameters $p^{(t+1)}(b_i|h_1)$ and $p^{(t+1)}(c_j|h_2)$ are easily obtained by marginalization and conditionalization from $p^{(t+1)}(h, w, \mathbf{b}, \mathbf{c})$.

The parameters $\psi(b_i, w), \psi(c_j, w)$ have no closed-form expression, due to the dependencies between

[2] We use h to denote (h_1, h_2) for brevity

the latent variables induced by the partition function $Z(\mathbf{b}, \mathbf{c})$. However, as is standard for conditional maximum entropy models like that of Equation 1, the expected complete data log-likelihood is maximized for the values of the parameter vector $\psi^{(t+1)}$ that satisfy the marginal constraints

$$\begin{aligned} q(w, b_i) &= p^{(t+1)}(w, b_i) \\ q(w, c_j) &= p^{(t+1)}(w, c_i) \end{aligned}$$

where

$$q(w, \mathbf{b}, \mathbf{c}) = p(w|\mathbf{b}, \mathbf{c}, \psi^{(t+1)})p(\mathbf{b}, \mathbf{c})$$

is the model-expected marginal count of w and $\mathbf{b}, \mathbf{c}$. The above equations can be solved iteratively for $\psi^{(t+1)}(w, b_i)$ with generalized iterative scaling (GIS) [6], which uses the following parameter update

$$\Delta\psi^{(t+1)}(w, b_i) = \frac{1}{C} \log \frac{p^{(t+1)}(w, b_i)}{q(w, b_i)}$$

where C is an upper bound on the number of terms in the exponent in Equation 1. Similar updates are used for $\psi(w, c_i)$.

3.1 Overrelaxed Updates and Regularization

In this section we discuss briefly two practical issues related to the learning algorithm.

First, both GIS and EM are known to be slow to converge, so that the doubly-iterative EM algorithm proposed in the previous section is impractically slow without modification. To make the algorithm practical, we use only a few GIS steps per EM step and then apply an adaptive, overrelaxed update [12]. For large models, this modified update gives convergence in under 75 iterations, whereas the standard EM update does not.

Secondly, while both latent variable models are intended to smooth a conditional distribution, they can also overfit their training data, in particular when the number of mixture components is large. Because of this, we sometimes use a deterministic annealing variant of EM [9, 14], where a free parameter corresponding to temperature is optimized on heldout data.

4 Experimental Evaluations

To evaluate our distributed latent variable model we first apply it to modeling bigrams, in order to compare it to single latent variable models. We then apply it to the more complex task of trigram modeling.

4.1 Modeling Verb Object Association

We begin with modeling the co-occurrence of bigrams composed of verbs followed by objects. The task is to obtain a model of the conditional distribution $p(o|v)$ of seeing an object given the preceding verb.

Our data come from a subset of the Penn Treebank Wall Street Journal Corpus [11]. This is a set of one million words of Wall Street Journal text, where each sentence is annotated with its syntactic structure in the form of a parse tree. The data are extracted from the parse trees by finding all right branching parent-modifier pairs where the parent is tagged as a verb and the child is tagged as a noun. Instances of the data then correspond roughly to verbs and objects, both direct and indirect, but also can include other nouns such as predicate nominals and the subjects in subject-verb inversions. The data contain 50,205 such verb-object pairs, and there are 5191 unique verbs and 8470 unique nouns, so the matrix is quite sparse. In order to avoid the problem of unseen words, we collapsed all low-count words into a single token. The count threshold for this normalization also affects the sparseness of the matrix, and we performed experiments which varied it.

The dataset was divided into 9-1 training-test splits, and we further divided the training set to give us held-out data on which to optimize model free parameters.

We use a bigram model smoothed by deleted interpolation as a baseline. This model gives the probability of an object given a verb as:

$$p_{BG}^{interp}(o|v) = \lambda_1 p_{ML}(o|v) + \lambda_2 p_{ML}(o) + \lambda_3 p_U \quad (2)$$

Here the weights λ_i are positive and sum to 1. The three distributions interpolated are the maximum likelihood bigram, unigram, and uniform distributions. Following Jelinek [10], we bin the histories based on frequency and set the weights to maximize the likelihood of heldout data.

Given a distributed model $p_{DM}(o|v)$ we interpolate it with the baseline by adding the term $\lambda_4 p_{DM}(o|v)$ to Equation 2 and determining the weights as above.

The baseline and distributed models are evaluated using data predictive perplexity. For a model $p(o|v)$,

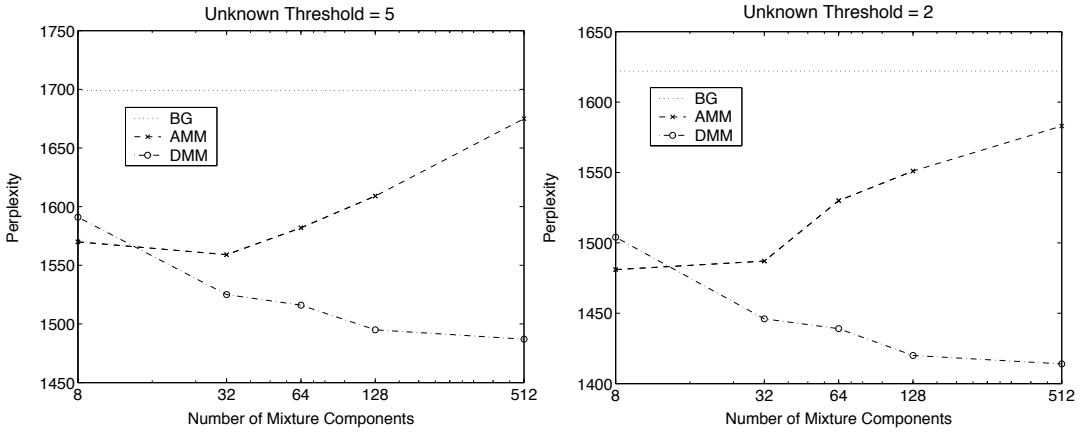


Figure 4: Perplexity Results for all three latent variable models for varying numbers of mixture components with two unknown word thresholds. An AMM with k mixture components has a latent class variable that takes on k values. A DMM with k mixture components has $log(k)$ binary latent variables

Table 1: Percentage improvement over the baseline smoothed bigram model for the best latent variable models of each type. UNK refers to the unknown word threshold.

% Improve over BG	UNK=2	UNK=5
AMM	8.6%	7.6%
DMM	12.8%	12.5%

the perplexity on a test set of verbs and objects is $\mathcal{P}_{\text{test}} = \exp\Big[-\frac{1}{n}\sum_{v,o} c(o,v)\log p(o|v)\Big]$.

Here $c(o,v)$ refers to the count of a verb-object bigram, and n is the total number of bigrams in the corpus.

Figure 4 gives perplexity results for the latent variable models with varying numbers of mixture components. The latent variable models are trained with annealed EM as described in Section 3.1, and the perplexities reported are for interpolated models as described above.

While the single latent variable model (AMM) does give better results for 8 mixture components, the distributed model consistently outperforms it for all other numbers of mixture components. It is also worthwhile to note that the AMM overfits its training data, even when annealed, while the distributed model shows no sign of overfitting, even when the number of mixture components becomes quite large. This is because the distributed model has a much more compact representation for the same number of mixture components. As mentioned before, the number of parameters in the distributed model scales logarithmically in the number of mixture components, but it scales linearly for the AMM.

As Table 1 shows, the best distributed models can improve the baseline by nearly 13%. In the next section, we examine the kinds of lexical "clusterings" discovered by the AMM and DMM.

4.1.1 Analysis of Lexical Co-occurrence Clusterings

One way to analyze the latent variable models is to examine the distributions $p(z|v), p(o|z)$ for the AMM and $p(\mathbf{b}|v), p(o|\mathbf{b})$ for the DMM. In the AMM, verbs v for which $p(\hat{z}|v)$ is high can be thought of as belonging to the $\hat{z}$ cluster. Similarly, objects o for which $p(o|\hat{z})$ is high can be thought of as belonging to the $\hat{z}$ cluster. Since the latent variable models here are mixture models, analyzing verbs and objects as belonging to one single cluster can hide some subtleties. Nonetheless, examining the parameters in this way provides us with useful insight into their representations of lexical co-occurrences

Table 2 shows one such "clustering" of the 200 most frequent verbs and 200 most frequent objects induced by the 32-class AMM on the verb-object data. For verbs and objects v and o, we assign them to the cluster $\hat{z}$ corresponding to $\hat{z} = \underset{z}{\text{argmax}}\, p(z|v)$ and $\hat{z} = \underset{z}{\text{argmax}}\, p(o|z)$ respectively. We display the top 8 clusters, sorted by the highest $p(z|v)$ value for verbs in the cluster.

Class	$p(z\|v)$	$p(o\|z)$
1	named	chairman, president
2	been, become, told, asked	issue
3	put, build	funds, system
4	did, do	it, business, something
5	are, used	issues
6	called, find, know	what
7	consider, approved, following	plan, bid, program, changes, proposal, bill
8	give, gives, giving, given, lost	support, right, money

Table 2: 8 classes induced by the 32 class AMM. The first column shows the verbs for which this class is the most probable. The second column shows the nouns which are most probable given this class.

Class	$p(\mathbf{b}\|v)$	$p(o\|\mathbf{b})$
1	said, says, were	executive
2	held, dropped, yield, rose, grew	point, %, cents, points
3	ended, ending, began, sent, reached	Nov., report, March, agreement, June
4	is, be, been, named, become	director, chairman, president, part, reason
5	closed	year, Tuesday, week, Wednesday, yesterday
6	set, consider, begin, rejected, made	acquisition, sale, plan, plans
7	told, asked, help, called, tell	us, investors, people, him, me
8	did, does, do, play, call, know	you, what, work, role, something

Table 3: 8 classes induced by the 5 variable DDM. The first column shows the verbs for which this class is the most probable. The second column shows the nouns which are most probable given this class.

Table 3 gives a similar clustering induced by the 5-variable DMM. The 8 clusters are chosen in the same fashion. Both models find some reasonable clusters of verbs and objects, but the AMM clusters are a little less sharply defined, and it induces some clusters that seem quite counterintuitive. The word "issue" is in a separate cluster from "issues," for instance.

4.2 Trigram Language Modeling

In this section, we evaluate our distributed latent variable model on trigram language modeling. Trigram models are widely used in speech recognition because they are easy to build and efficient to apply. Trigram language models also usually generalize in a straightforward way to longer contexts. Our data for this experiment also come from the treebank portion of the Wall Street Journal restricted to 10,000 distinct words. The dataset is described more completely in [17] and contains sections for training, optimizing free parameters parameters and testing. For these experiments, we do not anneal either model. Instead we regularize the models by stopping the EM algorithm when the models begin to overfit development data. For this data, annealing is not effective at regularizing the AMM, and annealing the DMM did not make a significant difference after interpolation.

Figure 5(a) gives perplexity results for distributed models with varying numbers of latent variables per word. It is natural to suppose that w_2 is the more effective predictor of w_3, and this is indeed shown in these experiments. In fact for all but the 10-bit case, it is always better to allocate all of them to the most recent word in the history. Even then, the models show little sign of saturating the amount of information present in either of the histories.

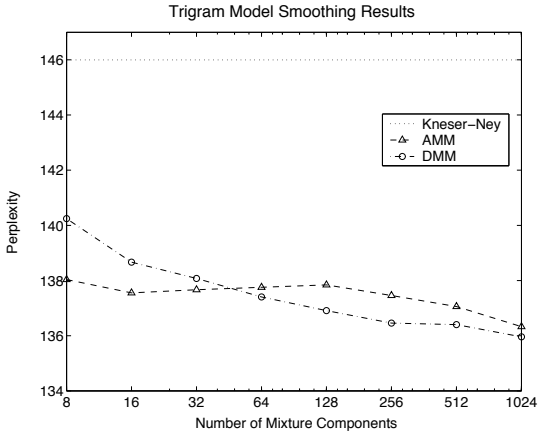
We also compared our model with an interpolated Kneser-Ney trigram model. The interpolated Kneser-Ney model is one of the most effective trigram language models, and Chen and Goodman demonstrate that a variant of this model consistently outperforms all other interpolation and backoff smoothing techniques [4] [3]. We use the Good-Turing estimate of the parameters of the Kneser-Ney model and we combine the AMM and DMM with it by linearly interpolating the two trained models. As before, we set the weights to maximize the likelihood of heldout data.

Figure 5(c) shows learning curves for the distributed model when interpolated with the trigram baseline described above. Again we compare with an aggregate Markov model. This time, the AMM is used only to smooth the distribution $p(w_3|w_2)$. As with the verb-object bigram results, we compare two models

[3] This variant, modified Kneser-Ney, can achieve a perplexity of 143.9 on this dataset.

(a) Perplexities of models with different numbers of latent variables

	$\mathbf{c}(w_1)$				
$\mathbf{b}(w_2)$	0	1	2	3	4
3	379	361	346	343	344
4	336	329	317	321	316
5	315	310	300	299	296
6	295	292	285	283	X
7	283	280	277	273	X
8	274	273	270	X	X
9	269	**261**	X	X	X
10	263	X	X	X	X



(c) Learning curves for DMM and AMM models with varying numbers of mixture components

Figure 5: Results of trigram language modeling experiments

with identical numbers of mixture components. For each bit count, we choose the DMM with the lowest heldout data perplexity. For all but 10 bits (1024 mixture components), this corresponds to allocating all bits to the most recent word. While the DMM slightly outperforms the AMM for larger numbers of mixture components, both models improve similarly (6.6 versus 6.8%) on the Kneser-Ney baseline. One explanation for the difference between these results and the verb-object bigram results is the difference in data sparseness. The AMM quickly overfits the verb-object data, but it is much harder to overfit the 10,000-word vocabulary trigram data.

5 Related Work and Discussion

Aside from Saul and Pereira [13], there is also other work on applying ideas similar to ours to language modeling. Wang et al. [15] investigate a language model based on what they call the latent maximum entropy principle. Like our distributed models, latent maximum entropy models are mixture models. Similarly, like our factored form from Equation 1, latent maximum entropy models are maximum entropy for a fixed posterior distribution over their hidden features. However, unlike our model, the Wang et al. [15] model is not distributed and uses a single latent variable.

The models which are most similar in spirit to ours are the connectionist models of Bengio [1] and Xu [17]. These models combine a neural network with a softmax output for probabilities. They are not sparse, and because of this, they must use $O(H \cdot V)$ time per epoch, where H refers to the number of unique histories and V refers to the vocabulary size. For most corpora, this number is prohibitively large. Finally, we recently introduced a much faster model which combines dimensionality reduction for continuous embedding of histories with a hierarchical mixture of experts for faster normalization [2].

Our distributed latent variable model is fully probabilistic, and its factored form allows it to avoid the prohibitively large $O(H \cdot V)$ calculation of the earlier connectionist models. Recall that the DMM requires time $O(L \cdot 2^{m+n})$. Since L is typically close to H (for our trigram data, L is 1 million and H is 600,000), this results in much faster training times when m and n are small. When m and n are large, though, the DMM also faces speed issues. The coupling of the latent variables forces us to sum over all mixture components when computing the expectations in the E-step. While this is possible for 1024-component mixtures, it becomes too large for more mixture components on larger corpora.

In terms of model form and inference, our model is closely related to the factorial hidden Markov model [8]. The factors of the factorial HMM correspond to our latent bits, but there are two major differences between the two models: The factorial HMM is a generative model in which the observed variables at a time slice are generated from the factors. By contrast, our DMM assumes that the the latent bits are distributed conditioned on a history. Furthermore, the factorial

HMM explicitly models the dependencies between hidden variables at different time slices, whereas we do not. Ghahramani and Jordan give a mean-field approximation to the posterior for their factorial HMM. Unfortunately, the introduction of the mean field parameters in our model recouples h and w, resulting in exactly the $O(H \cdot V)$ computation we avoided by specifying its factored form.

6 Conclusion

We presented a distributed latent variable model for lexical co-occurrence data. Using the framework of graphical models, we derived an EM algorithm and compared the learned models with the single latent variable aggregate Markov model [13]. The result was that the distributed latent variable model significantly outperforms the single latent variable AMM and is able to improve significantly on both bigram and trigram baselines.

As mentioned in the previous section, our model requires time exponential in the number of latent variables in order to perform exact inference. For 9 binary latent variables, this is not large, but for models with many variables, this number quickly becomes too large. This is due to the directed nature of our graphical model. It is worth mentioning that it is also possible to use undirected graphical models to represent the essential conditional independences expressed in this model. We are currently investigating such models together with approximate inference techniques [18] to make learning and inference practical.

Acknowledgements

We would like to thank Lawrence Saul and Naftali Tishby for valuable discussions throughout the course of this work. We thank Peng Xu for help preprocessing the data for the trigram language modeling experiments. Blitzer and Pereira are partially supported by DARPA under SRI subcontract NBCHD030010.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] J. Blitzer, K. Weinberger, L. Saul, and F. Pereira. Hierarchical distributed representations for statistical language modeling. In *Advances in Neural Information Processing Systems 17*, 2004.

[3] P. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–311, 1993.

[4] S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL*, 1996.

[5] M. Collins. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[6] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematic Statistics*, 43:1470–1480, 1972.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[8] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Mach. Learn.*, 29(2-3):245–273, 1997.

[9] T. Hofmann and J. Puzicha. Statistical models for co-occurrence data. Technical Report AIM-1625, MIT, 1998.

[10] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1997.

[11] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[12] R. Salakhutdinov and S. Roweis. Adaptive overrelaxed bound optimization methods. In *Proceedings of ICML*, 2003.

[13] L. Saul and F. Pereira. Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of EMNLP*, 1997.

[14] N. Ueda and R. Nakano. Deterministic annealing variant of the EM algorithm. In *Advances in Neural Information Processing Systems 7*, 1995.

[15] S. Wang, D. Schuurmans, F. Peng, and Y. Zhao. Semantic n-gram language modeling with the latent maximum entropy principle. In *Proceedings of AISTATS*, 2003.

[16] P. Xu, A. Emami, and F. Jelinek. Training connectionist models for the structured language model. In *Proceedings of EMNLP*, 2003.

[17] P. Xu and F. Jelinek. Random forests in language modeling. In *Proceedings of EMNLP*, 2004.

[18] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *IJCAI 2001 Distinguished Lecture track*.